

DEPARTMENT OF HEALTH AND HUMAN SERVICES

Food and Drug Administration

[Docket No. 97D-0282]

DMB

Display Date	01-10-02
Publication Date	01-11-02
Officer	A. Corbin

Medical Devices: General Principles of Software Validation; Final Guidance for Industry and FDA Staff; Availability

AGENCY: Food and Drug Administration, HHS.

ACTION: Notice.

SUMMARY: The Food and Drug Administration (FDA) is announcing the availability of the guidance entitled "General Principles of Software Validation." This document provides guidance to medical device manufacturers and FDA staff concerning requirements for validating software used within medical devices, in device production, or in implementing the manufacturer's quality system.

DATES: Submit written or electronic comments at any time.

ADDRESSES: Submit written requests for single copies on a 3.5" diskette of the guidance document entitled "General Principles of Software Validation" to the Division of Small Manufacturers, International and Consumer Assistance (HFZ-220), Center for Devices and Radiological Health (CDRH), Food and Drug Administration, 1350 Piccard Dr., Rockville, MD 20850. Send two self-addressed adhesive labels to assist that office in processing your request, or fax your request to 301-443-8818. See the **SUPPLEMENTARY INFORMATION** section for information on electronic access to the guidance.

Submit written comments concerning this guidance to the Dockets Management Branch (HFA-305), Food and Drug Administration, 5630 Fishers Lane, rm. 1061, Rockville, MD 20852. Submit electronic comments to <http://www.fda.gov/dockets/ecomments>. Comments are to be identified with the docket number found in brackets in the heading of this document.

FOR FURTHER INFORMATION CONTACT: John F. Murray, Center for Devices and Radiological Health (HFZ-340), Food and Drug Administration, 9200 Corporate Blvd., Rockville, MD 20850, 301-594-4659.

SUPPLEMENTARY INFORMATION:

I. Background

This final guidance document entitled "General Principles of Software Validation" provides guidance to medical device manufacturers and FDA staff concerning requirements for validating software used within medical devices, in device production, or in implementing the manufacturer's quality system. It replaces the draft guidance that FDA issued for comment on June 9, 1997, and published in the **Federal Register** of July 25, 1997 (62 FR 40099).

We received responses from 36 organizations and individuals, with more than 650 questions, comments, and specific recommendations for changes to the guidance. However, further work on the guidance was interrupted by other high priority activities, including implementation of the Food and Drug Administration Modernization Act of 1997, FDA's response to year 2000 software concerns, and two rounds of implementation of our first medical device performance standard. Because of the delay in issuing this final guidance, we have chosen to summarize our response to the comments received. As with any guidance, we will continue to accept comments and may update this document in the future.

The following summarizes the comments we received, and significant changes we made to the guidance in response to those comments:

A. Intended Scope

From a few of the comments received, it appears that some parties may not have realized the full breadth of the quality system regulation. The software validation requirement in 21 CFR 820.70(i) of the quality system regulation also applies to automated tools used to design medical devices and tools used to develop software. Since the first medical device good manufacturing

practice regulation was published in 1978, there has always been an explicit validation requirement for software used in device production or used to implement the quality system. When design controls were introduced into the quality system regulation in 1997, that software validation requirement was extended to software used to design devices, such as computer-aided design and software development tools. FDA clearly addressed this issue at the end of its response to comment 136 in the preamble to the quality system regulation (61 FR 52602 at 52630, October 7, 1996). A copy of the text is included at the end of this section.

Some comments objected to the discussion of validation activities during the predesign “concept” phase of software development, both because the quality system regulation does not apply to research activities, and because there is too little information available at that point to make any validation related activity worthwhile. In response to these concerns, we have removed all reference to validation activities during the “concept” phase.

Other comments noted that the guidance covered more than just validation issues, and suggested changing the title to broaden the scope of the guidance. We acknowledge that the scope of the guidance is somewhat broader than the scope of validation in the strictest definition of that term. However, we have chosen not to change the title of the guidance. Planning, verification, testing, traceability, configuration management, and many other activities discussed in the guidance are important activities that together help to support a final conclusion that software is validated.

Some comments expressed concerns that the guidance might be applied too rigorously by FDA investigators, and some pharmaceutical manufacturers raised questions about how the guidance would be applied to their drug manufacturing operations. The agency’s good guidance practices (GGPs) clearly state the role of FDA guidance. Alternative approaches that accomplish full compliance with the quality system regulation are acceptable. While it is clearly intended for medical device manufacturers, the guidance may also be useful to the pharmaceutical industry and other industries regulated by FDA.

Many comments suggested that we move all discussions regarding use of off-the-shelf (OTS) software to the agency's guidance entitled "Off-the-Shelf Software Use in Medical Devices." In response to these comments, specific cross references to that document have been added within the text of this guidance. However, the OTS guidance document deals specifically with premarket submissions for OTS software contained in medical devices. It is not the appropriate guidance for OTS software used in manufacturing and quality systems applications.

B. Flexibility

Numerous comments cited overly restrictive language and lack of sufficient implementation flexibility in the draft guidance. For example, many comments noted that the guidance implies use of a "waterfall" as the preferred life cycle development methodology. Several comments suggested that more discussion was needed regarding "rapid application development" and "component-based methodologies," as well as "build a little/test a little" as an acceptable methodology. Other comments asked for specific examples of available life cycle models that could be used. In response to these comments, and in accordance with our own GGP's, we have carefully rewritten the text to remove any direct or implied use of the words "shall" or "must," except where we describe or reference a regulation. We also have added language to specifically state that incremental development methodologies may be used, and that activities and tasks can be performed in a different order, if called for by the chosen life cycle model. However, for ease of description, we have retained an organization of activities based on "requirements," "design," "coding (or construction)," and "testing." Regardless of the order in which tasks are accomplished, these four categories of activities are common to most life cycle models. We have not included examples of the dozens of life cycle models that are available. To do so could imply agency endorsement of certain life cycle models that are included over those models that are not included. Instead, you are referred to many of the textbooks and other references listed at the end of the guidance, which provide details of many of these life cycle models.

One group of comments objected to any use of the word “all” when describing items to be included in specification documents, noting that “all” is not a quantifiable term. Other comments suggested use of the word “may” rather than “should.” On the other hand, a few comments asked for a specific compliance matrix, so that manufacturers would know exactly how to comply with FDA expectations. We have not adopted these suggested changes. We believe that agency guidance should identify and encourage use of approaches known to have been used effectively, while the manufacturer retains the prerogative to choose alternative approaches that are equally effective. Based on variables such as firm size and structure, device risk, project size, and complexity, manufacturers have the flexibility to choose different approaches for different projects, and to select effective approaches that best fit their specific needs.

C. Format

Several comments suggested use of the framework and format in international guidelines such as ISO 9000–3, GAMP, IEEE Software Standards and ISO/IEC 12207. We have drawn information from each of these sources and many other listed references, but unfortunately, there is no single format available. We have rewritten the guidance to address specific suggestions for wording changes and simpler language. Some comments asked for extensive use of charts, analogies, and examples for the concepts presented in the document. While valuable, such an approach could easily triple the size of the guidance. Instead, we suggest referring to any of the extensive list of references included at the end of the guidance for more details on specific implementation approaches.

D. Differences Between Hardware and Software

Regarding the discussion of differences between hardware and software, the comments were somewhat divided. Some comments applauded the agency for recognizing the legitimate differences between hardware engineering and software engineering. Other comments argued that “software is not different” and suggested deletion of all or most of this section, either because it was

unnecessary, or because it could be misinterpreted by software developers who lack sufficient engineering discipline. One comment suggested emphasizing the similarities of the engineering discipline needed to build both hardware and software. We have chosen to keep this section because we believe it explains part of the rationale for why software must be thoroughly validated, and why the software development process needs to be carefully controlled and managed. We have also added additional information regarding the impact of mobility of software professionals on the long-term maintenance of software and the need for thorough documentation.

Some comments objected to the discussion of standardization and reuse of software components and asked for more recognition of the trend toward increased use of OTS and component-based development methods. Other comments objected to the statement that “repairs made to correct software defects establish a new design.” We have revised the text to address both of these concerns.

E. Principles of Software Validation

We reorganized and rewrote the section regarding “Principles of Software Validation” to address the comments received. For example, we moved the subsection dealing with documenting software “Requirements” to the front of the section to reflect the importance of requirements in the validation process. We clarified language regarding “predetermined” requirements to allow for incremental or evolutionary development of requirements during the development project. However, we have retained the concept that documented requirements should be established prior to formal testing or other verification activities to provide “objective” evidence that those requirements were met.

The subsection previously entitled “Testing” is retitled “Defect Prevention” and is revised to emphasize the importance of preventing software defects, as opposed to trying to “test quality into” software.

We have renamed the subsection on “Timing.” In response to several comments concerning validation continuing “for the entire life cycle,” we have rewritten the text, but have retained the

concept. At each stage of the software life cycle, there is information available that can contribute to a conclusion that the software meets user needs and intended uses. Therefore, the validation process does not end when the device is shipped.

We replaced the subsection on “Management” with a new subsection dealing with the “Software Life Cycle.”

We have clarified the subsections dealing with “Plans” and “Procedures” to distinguish between plans that define what to do, and procedures that describe how to do it.

The subsection entitled “Partial Validation” is substantially rewritten and retitled “Software Validation After a Change.” Many readers misinterpreted the statement that “software cannot be partially validated” and thought we intended all validation testing to be repeated every time any change is made. That is not what we meant. Based on the comments received, we have rewritten the discussion to emphasize the need for regression analysis after a change, followed by an appropriate level of regression testing to reestablish the validation status of the software. We have deleted specific discussion of retrospective validation and reverse engineering of nonvalidated software, but these issues should be covered during the regression analysis.

We have retitled and rewritten the subsection on “Amount of Effort.” Now titled “Validation Coverage,” it still describes an approach that ties the level of validation and verification effort to the safety risk and complexity of the software.

We revised the subsection on “Independence of Review” to provide greater flexibility and a better explanation of its intent.

The subsection previously entitled “Real World” is now entitled “Flexibility and Responsibility,” and reemphasizes that device manufacturers/software developers have a lot of flexibility in how they implement their software validation process, but the device manufacturer is ultimately responsible for the adequacy and effectiveness of the selected approach.

F. Terminology

Some of the most significant comments we received had to do with our basic definition of software validation. In the previous draft guidance, we relied upon technical definitions used by the National Institute of Standards and Technology and by the Institute of Electrical and Electronic Engineers. These technical definitions created some confusion with other definitions in our quality system regulation. Numerous comments objected to our use of “validation” as an umbrella term to cover “design review” and “verification” as well as validation. They stated that both design review and verification are distinctly separable quality concepts and are not a part of validation. In response to these concerns, we have changed the definition of software validation to be more consistent with the quality system regulation and other international quality standards. Our revised definition of software validation is derived directly from the definitions of “validation” and “design validation” in the quality system regulation.

Comments also objected to the title “Typical Validation Tasks” at the end of each subsection in the section V of the guidance and suggested that they are really verification tasks. Other comments objected to possible interpretation of these as mandatory tasks. In response to these comments, we have also added text to explain that there are typical verification and testing tasks that support an overall conclusion that software is validated. Thereafter, when we discuss “Typical Tasks Supporting Validation,” we do not try to differentiate between verification tasks versus validation tasks. Instead, we have revised the text to list “Typical Tasks.” While we want to avoid any inference that the tasks are mandatory in every case, the guidance makes the point that these are “typical” approaches that are recommended by software engineering standards and textbooks, and widely used by many software engineering professionals.

Several comments noted inconsistencies in terminology from that contained in the quality system regulation, in two software guidances issued by the Office of Device Evaluation, and in the FDA glossary of computerized system and software development terminology. These comments also suggested use of the term “risk analysis” instead of “hazard analysis” throughout the software

validation guidance. We have revised the guidance to incorporate the term “risk analysis” throughout. However, we continue to emphasize that while there are many different risks (e.g., economic or time to market), FDA is concerned about safety risk (hazard). At their next revision, we expect to update other software guidance documents and the FDA glossary with consistent definitions of validation, verification, and risk analysis. In addition, we now use the term “user site testing” rather than “installation testing” to describe testing performed at the user site and outside the control of the software manufacturer.

Some comments questioned whether OTS software could be validated because the device manufacturer frequently does not have access to the source code. These comments suggested that OTS software should be “qualified” rather than “validated.” However, we believe that the evidence developed by a device manufacturer concerning OTS software is a true validation because it directly supports a conclusion that the software meets user needs and intended uses. Where the source code is not available, it is incumbent upon the device manufacturer to use other means (such as audits, or more extensive black box testing) to infer the structural integrity of the OTS software. This issue is clearly addressed in comment 136 of the preamble to the quality system regulation (61 FR 52602 at 52630).

Other comments from the pharmaceutical industry suggested incorporation of widely understood process validation terminology (i.e., installation qualification (IQ), operational qualification (OQ), and performance qualification (PQ)) to describe software validation. Another comment suggested use of “product performance qualification” rather than “design validation.” We have added a section that refers to the various types of qualification, but we have chosen not to adopt “qualification” terminology in explaining software validation requirements. Of course, manufacturers may continue to organize their validation efforts using IQ/OQ/PQ terminology, if they wish.

In response to comments, a new subsection has been added to explain the differences between “requirements,” which may be general in nature, versus “specifications,” which are developed to an engineering level of detail.

Several comments objected to use of undefined terms such as “microcode” and “assertions.” We reiterate that these and many other terms used throughout the guidance are specifically defined in the FDA glossary of computerized system and software development terminology, which is available at http://www.fda.gov/ora/inspect__ref/igs/gloss.html.

G. Design Review

As noted above, design reviews are not a part of validation. In fact, several comments noted that results of verification and validation are inputs to design reviews—not the other way around. To emphasize this point, we moved the subsection on “Design Reviews” outside the section on “Typical Tasks Supporting Validation.” We also added information about the difference between formal design reviews that are mandated by the quality system regulation versus less formal technical reviews.

H. Traceability

A few comments objected to the guidance regarding “traceability analysis,” especially the discussion at the end of the subsection on “Coding.” Two comments noted that for very complex programs with thousands of lines of code or thousands of modules, the traceability analysis would be extremely complex and of little value. One suggested that design review was an adequate substitute for traceability analysis. We disagree. Traceability is an essential aspect of verification, and it is an important input into design reviews. We therefore do not believe that design review could be an adequate substitute for traceability analysis.

One comment stated that requirements are not always neatly structured, and it is very difficult to trace exactly how they are implemented in the design. There are numerous many-to-one and one-to-many relationships to be mapped from requirements to design to code. We agree with this

observation; however, it actually further supports the need for traceability. The larger and more complex the project, the more important the traceability analysis becomes. Therefore, we have retained the discussions regarding traceability, and in response to several other comments, we have added traceability of software requirements to the safety risk analysis.

Another comment noted that inherent traceability can be built into documentation and code without having to have a separate traceability document. We agree and for that reason have avoided use of the most commonly used term—"traceability matrix." Three common approaches are traceability matrix, using computer databases to evaluate traceability, or building inherent traceability into the structure of the documentation and code. There may be many other approaches to traceability. Software developers have flexibility in how they want to implement traceability.

I. Risk Analysis

Many comments questioned the concept of a software failure modes and effects analysis (FMEA). They stated that given the difficulty of predicting specific software failure modes, FMEA is better used as a system level risk analysis tool. We have revised the guidance to discuss software risk analysis within the context of system safety. However, while we acknowledge some limitations in its use, we also believe that software FMEA can be a useful tool, especially for safety critical aspects of software applications. It may also be useful early in the development process for analyzing safety critical software requirements.

One comment objected to the suggestion that risk analysis begin at the stage where requirements are defined. However, to be useful and have an impact on the software development process, we believe that risk analysis needs to begin early and needs to be updated as the project progresses. In addition, we have revised various portions of the guidance to emphasize that the level of safety risk is a major factor in determining the level of effort to be applied in testing and other verification and validation tasks.

J. Planning

In response to comments, we have changed the subsection on “Management” to be entitled “Quality Planning.” It now provides a more general discussion of the software validation and verification concerns to consider during quality planning.

Several comments questioned the idea of early test planning, which was recommended in the draft guidance. For example, they argued that there is insufficient information available during requirements development to be able to develop a system test plan or an acceptance test plan. We disagree and have retained the recommendations for early test planning, but we have specified that test plans and test cases should be created as early in the software development process “as feasible.” One of the important criteria, both for requirements and for design, is that they be testable. The fact that there is insufficient information for a particular test plan is valuable feedback to the development process that perhaps the requirements or design processes are not yet sufficiently complete. Planning is a dynamic activity that should be reexamined and updated as the project progresses.

K. Requirements

Many comments objected to use of the word “all” in describing what is typically specified in software requirements. We agree that requirements frequently do not specify “all” that they should. However, that is widely recognized as one the major flaws in software development, and its correction is one of the most important messages intended by this guidance. In order to be complete, a software requirements specification should cover all the pertinent issues—not just a selected few.

One comment noted that requirements may not always be measurable. We have changed the text to state that requirements should be “measurable or objectively verifiable.”

A few comments noted that “internal interfaces” and “all ranges of values the software will accept” are a part of design—not requirements. We agree regarding internal interfaces and have changed the text accordingly. However, since software requirements are derived from system

requirements, there may be some internal system interfaces prescribed from the high level system design that would impact software requirements. Regarding “ranges of values,” we note that there is rarely a bright line of demarcation between requirements and design. Software developers have flexibility as to where in their life cycle they wish to cover particular issues. We rejected most comments requesting even greater levels of detail and specificity regarding static verification techniques. For example, several comments asked for more detail regarding “requirements evaluation” and “interface analysis.” Details on these techniques are available in many of the references listed at the end of the guidance. FDA investigators will expect to see a verification procedure that includes a means for identifying and resolving incomplete, ambiguous, and conflicting requirements, as required by the regulation. They will also expect to see objective documented evidence that the verification procedure was implemented.

L. Design

We have retained wording about the need for design specifications to be complete enough for programmers not to have to make ad hoc decisions. The intent is to ensure that the code created is consistent with the design specification. When programmers or engineers decide to add new functionality not identified previously in the requirements or design, those specifications need to be updated to reflect the actual code created. The project manager, design team, and any future maintainers of the software need to have accurate documentation in order to do their work.

We have dropped the listing of specific approaches to software design, and we have included a more general description of what should be included in a software design specification. Some comments considered the previous list to be too prescriptive as well as incomplete.

We recognize that portions of the software are completed and released incrementally, and life cycle processes are repeated iteratively. The intent is that those portions of the software have design documentation that is consistent with the software application that is implemented. One comment noted that in a rapid application development (RAD) environment, there is typically no formal design document in place during coding. We recognize that RAD is valuable as a

prototyping tool, but its use does not preclude the need to document the specific design, once it is agreed upon.

M. Coding

We have changed the title of this subsection to reflect that the creation of a software application can be either through coding, or through combining existing software components, such as OTS software products or functional components from existing code libraries.

Comments objected to the idea of having to keep results of all compilations of the code. In response, we have revised the discussion of compiler error checking to state that the results of the “final” compilation of the code should be retained to document any errors that remain uncorrected in the final software product.

N. Testing by the Software Developer

We renamed and revised this subsection to provide a better explanation of the purpose of testing, and to avoid prescriptive language concerning use of specific testing techniques. We have added language regarding use of incremental development and testing methodologies. We expanded the discussion of testing coverage to explain how different degrees of coverage should be considered for varying levels of risk, and that the manufacturer has flexibility to choose the right level of coverage.

One comment noted that the intent of testing is to find errors, and suggested a better explanation of this and other tenets of a software testing strategy. We have added such an explanation.

Other comments argued that statistical testing based on usage profiles is more effective than extensive structural testing in finding software defects. We agree that statistical testing is one of many valuable testing methodologies, and we have added information about its use. However, it is important to note that statistical testing is an adjunctive approach, rather than an outright replacement for other types of testing.

O. User Site Testing

Based on several comments, we have renamed the subsection formerly entitled “Installation Testing” and moved it into the section on life cycle activities. User site testing can be any one of several types of testing performed by the user or by others at the user site. System level testing performed by the software developer under conditions that simulate the user’s environment is an important part of validation for some products, and it may substitute for some aspects of user site testing. However, for certain products such as blood establishment software, there are specific FDA requirements for additional testing to be performed at the user site. For manufacturing and quality system software, user site testing is frequently performed by the device manufacturer.

P. Maintenance and Software Changes

Several comments objected to the statement that “all modifications are design changes,” noting that some changes, such as a correction of coding errors, do not change the intended design. We have made appropriate changes to the text. However, we continue to emphasize that the validation of all software changes needs to include a regression analysis and, as appropriate, regression testing to show that the change has not negatively impacted the software.

In response to other comments, we have added information regarding anomaly evaluation, problem identification and resolution tracking, and the need to update documentation.

Q. Process and Quality System Software

We have added a new section to the document dealing with validation of automated process equipment and quality system software. This change was in response to the many comments that raised issues and asked for more detailed information about validating such software, especially OTS automated equipment and OTS software.

Many comments discussed the difficulties encountered in trying to validate OTS software, and suggested a different approach for validation of manufacturing and quality system software. Source code and life cycle documentation are frequently unavailable for review, so structural testing

is usually not possible. Auditing the vendor's software development activities is one possibility, but some software vendors will not agree to being audited. One comment suggested that risk analysis, design, coding, and unit testing should not apply to quality system software, especially if it is purchased, and further suggested that functional testing is the most that can be expected. Several comments suggested that for widely used applications, there can be a reasonable assumption that the vendor validated the software at the time it was developed, and that installation qualification by the user should be sufficient. Many of these issues are addressed in the response to comment 136 in the preamble of the quality system regulation (61 FR 52602 at 52630).

It is not the agency's intent to discourage use of OTS computer products. The activities described in the guidance can be shared between the vendor and device manufacturer (the user). However, we believe that the principles and activities described in the guidance are important for an overall conclusion that software is validated for its intended use. Device manufacturers are required to have purchasing controls for the products and services they receive. Such controls are an important part of decision making regarding OTS software. Our experience is that "assumptions" regarding validation by the vendor are not always well founded. Each OTS software product needs to be individually evaluated based on the intended use of the software, available life cycle documentation, available verification and validation evidence, and most importantly the device safety risk posed by the automated process. Device manufacturers can use multiple sources of information, but are ultimately responsible for documenting the basis for their conclusion that the software is validated for its intended use.

Several comments suggested alternative approaches for certain types of software, such as operating systems and certain tools used in software development, such as compilers and robust "middleware" such as Oracle, Documentum, or Lotus Notes. We have added suggestions for alternative approaches, while still retaining the basic requirement that the software must be validated for its intended use.

A few comments questioned who is responsible for validation of OTS software. One questioned FDA's authority to regulate software vendors, but argued that device manufacturers cannot be responsible because they lack access to source code and life cycle documentation. Another noted that vendors frequently change their hardware and software, resulting in unreasonable FDA expectations for revalidation of each change. One comment asked for more details regarding the impact of the supplier's quality system on purchasing decisions. In response to these comments, we reaffirm that FDA holds the device manufacturer responsible for the software validation requirement. This responsibility can be further delegated in part through contracting and purchasing controls, and monitored through supplier audits or other means, but the device manufacturer is ultimately responsible for its decision to choose a particular software product. The fact that a vendor refuses to provide access to its development process or documentation does not relieve the device manufacturer of this responsibility. Likewise, we note that the device manufacturer is not obligated to install every software upgrade offered by a vendor. Validation of those upgrades and support from the vendor, including access to the necessary vendor documentation, need to play an important role in the upgrade decision.

Some comments argued that software validation should be treated more like process validation, which is only required if the output of the process cannot be fully verified by subsequent inspection and testing. Other comments asked for clarification of the term "verification by output" and asked whether it negated the requirement for software validation. One comment argued that output of software driven systems can never be fully verified. Another comment suggested the consideration of intended use and dependence upon software for proper operation of the process to determine whether verification could be substituted for software validation.

In response to these comments, we believe there are very few examples where "verification" in lieu of software validation could be justified, and even in those cases, most manufacturers would choose to validate the software rather than go through repeated verifications of output. For example, while every aspect of a drawing from a computer-aided design (CAD) system can be independently

verified, no user of a CAD system is likely to go to that trouble or expense for every aspect of every drawing. Likewise, because software itself cannot be fully verified, automated software development tools used to create medical device software must be validated for their intended use.

Requirements are needed to establish intended use, the degree of dependence on the software, and therefore the degree of validation needed. The device manufacturer decides whether or not to use OTS software. The ability to validate for intended use and vendor support for the effort should be a part of that decision. Static analysis and structural testing are techniques to be used in evaluating source code and life cycle documentation, when these items are available. Otherwise, the device manufacturer is dependent upon functional testing alone. This issue is discussed in response to comment 136 in the preamble to the quality system regulation (61 FR 52602 at 52630). The impact on the safety and quality of the medical device is an important determining factor in the approach and level of effort to be applied for validating automated manufacturing and quality system software, just as it is for software in a medical device.

R. References

There were numerous recommendations for additional references. Those and many other reference books, international standards, and FDA guidance documents have been added to the appendix at the end of the validation guidance.

For ease of cross reference, the text of comment 136 from the preamble of the quality system regulation is included below:

136. One comment on § 820.70(h), "Automated processes," (now § 820.70(i)), stated that the section should be revised to reflect that software used in such systems must be validated for "its intended use," not simply validated. Another comment stated that most companies buy software currently available on the market and do not make changes to the software. It was recommended that § 820.70(h) allow for use of outside personnel for validation runs and not necessarily require the development of a software validation procedure. One comment suggested that the section should allow verification rather than

validation of off-the-shelf software. Several comments on “automated processes” stated that the term “data processing systems” was unclear and its inclusion rendered the requirement too broad. Others asked for clarification of “automated data processing systems.”

FDA has modified the requirement to mandate validation for the intended use of the software. In addition, the requirement that the software be validated by individuals designated by the manufacturer has also been deleted to make clear that validation may be performed by those other than the manufacturer. However, whether the manufacturer designates its own personnel or relies on outside assistance to validate software, there must be an established procedure to ensure validation is carried out properly.

FDA has maintained the requirement for validation because the agency believes that it is necessary that software be validated to the extent possible to adequately ensure performance. Where source code and design specifications cannot be obtained, “black box testing” must be performed to confirm that the software meets the user’s needs and its intended uses.

FDA emphasizes that manufacturers are responsible for the adequacy of the software used in their devices, and activities used to produce devices. When manufacturers purchase “off-the-shelf” software, they must ensure that it will perform as intended in its chosen application.

FDA has amended the requirement to state “When computers or automated data processing systems are used as part of production or the quality system,” for clarification. Software used in production or the quality system, whether it be in the designing, manufacturing, distributing, or tracing, must be validated.

II. Significance of Guidance

This guidance document represents the agency’s current thinking on software validation. It does not create or confer any rights for or on any person and does not operate to bind FDA or the public. An alternative approach may be used if such approach satisfies the applicable statutes and regulations.

The agency has adopted GGPs, and published the final rule, which set forth the agency’s regulations for the development, issuance, and use of guidance documents (21 CFR 10.115). This guidance document is issued as a level 1 guidance in accordance with the GGP regulations.

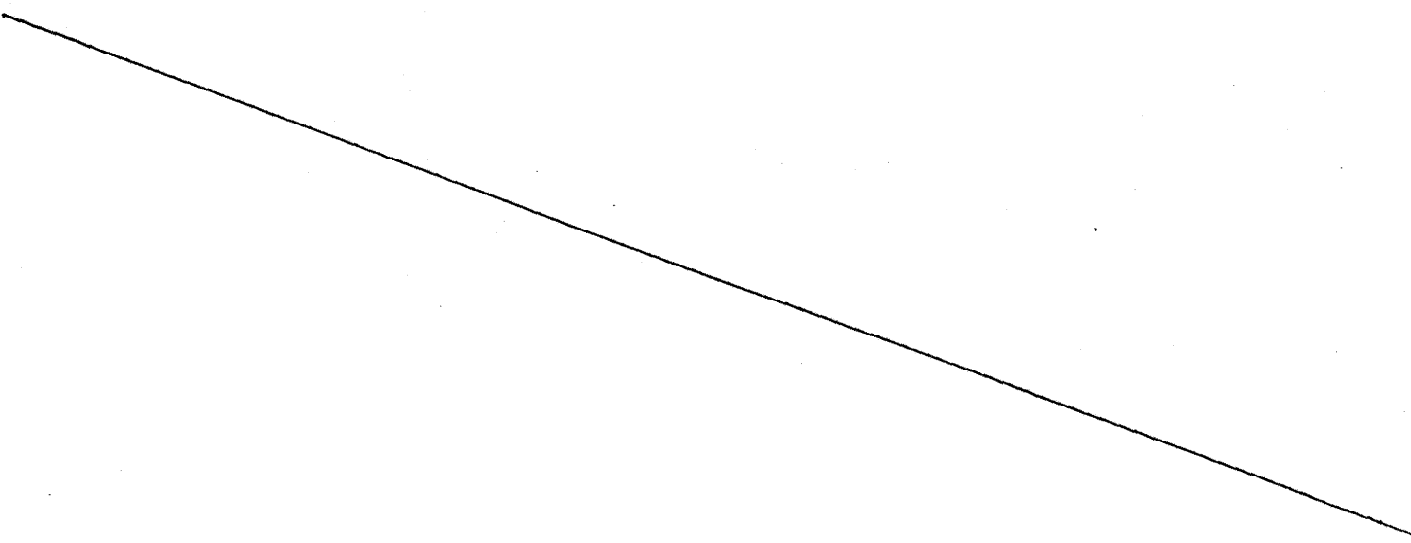
III. Electronic Access

In order to receive "General Principles of Software Validation" via your fax machine, call the CDRH Facts-On-Demand system at 800-899-0381 or 301-827-0111 from a touch-tone telephone. Press 1 to enter the system. At the second voice prompt press 1 to order a document. Enter the document number (938) followed by the pound sign (#). Follow the remaining voice prompts to complete your request.

Persons interested in obtaining a copy of the guidance may also do so using the Internet. CDRH maintains an entry on the Internet for easy access to information including text, graphics, and files that may be downloaded to a personal computer with Internet access. Updated on a regular basis, the CDRH home page includes the civil money penalty guidance documents package, device safety alerts, **Federal Register** reprints, information on premarket submissions (including lists of approved applications and manufacturers' addresses), small manufacturers' assistance, information on video conferencing and electronic submissions, Mammography Matters, and other device-oriented information. The CDRH home page may be accessed at <http://www.fda.gov/cdrh>. Guidance documents are also available on the Dockets Management Branch Internet site at <http://www.fda.gov/ohrms/dockets/default.htm>.

IV. Comments

Interested persons may submit to the Dockets Management Branch (address above) written or electronic comments regarding this guidance at any time. Submit two copies of any comments,



except that individuals may submit one copy. Comments are to be identified with the docket number found in brackets in the heading of this document. The guidance document and received comments may be seen in the Dockets Management Branch between 9 a.m. and 4 p.m., Monday through Friday.

Dated: 12/11/01
December 11, 2001.

Linda S. Kahan
Linda S. Kahan,
Deputy Director,
Center for Devices and Radiological Health.

[FR Doc. 01-²???? Filed ??-??-01²; 8:45 am]

BILLING CODE 4160-01-S

CERTIFIED TO BE A TRUE
COPY OF THE ORIGINAL

Regina L. Cole